OULUN SEUDUN
AMMATTIKORKEAKOULU

Niko Tauriainen

QT-SOVELLUSKEHITYSTYÖKALUT ANDROIDILLE

QT-SOVELLUSKEHITYSTYÖKALUT ANDROIDILLE

Niko Tauriainen
Opinnäytetyö
Kevät 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Niko Tauriainen
Opinnäytetyön nimi: Qt-sovelluskehitystyökalut Androidille
Työn ohjaaja(t): Eero Nousiainen
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 38 + 2 liitettä

Tämän opinnäytetyön tarkoituksena oli selvittää Qt-koodin toimivuutta Android-laitteessa Kuru Digital Creations Oy:lle. Työn jälkeen tuli pystyä sanomaan, toimiiko Qt-koodi tarpeeksi hyvin Androidissa, jottei yhtiöön tarvittaisi erillistä Android-osaajaa, ainakaan nykyisten sovellusten käännöksiin.

Jo ennalta tiedettiin olemassa olevasta Necessitas-projektista, joka luo Qt-porttausta Androidia varten ja päätettiin työskennellä tuon projektin tuomien työkalujen avuin. Opinnäytetyön tavoitteena oli saada toimimaan Kuru Digital Creations Oy:n kehittelemä Scouting-sovellus Android-laitteessa. Sovellus on toteutettu kokonaan Qt C++:lla ja Qt Quickillä. Sovelluksella on tarkoitus pystyä tekemään kykyjenetsijöille hyödyllisiä toimintoja, kuten pelaaja-arviointeja.

Työssä käytettiin ketterää projektinhallintamenetelmää Scrumia projektin läpivientiin. Projekti sisälsi useampia pienempiä projekteja, joilla testailtiin Qt:n eri osa-alueiden toimivuutta Androidissa.

Scouting-sovelluksen tärkeimmät osa-alueet tämän projektin kannalta saatiin toimimaan Android-laitteessa. Sovellus on kuitenkin vielä kehitysvaiheessa. Työn pienempien projektien ja Scoutingin toimivuuden tuloksista voidaan nähdä, että Qt-koodia voidaan ajaa Necessitasin avulla Android-laitteessa ilman liian isoja ongelmia. Näiden tulosten perusteella voidaan todeta, että erillistä Android-osaajaa ei tarvita, ainakaan nykyisissä projekteissa.

Asiasanat:
Qt C++, Qt Quick, Android, Necessitas, Qt Lighthouse, tablet-laitteet

SISÄLLYS

LYHENTEET JA TERMIT

CSS	Cascading Style Sheets. Tyylikieli, jolla määritellään esimerkiksi HTML-dokumentin ulkoasu ja esitystapa.
D-Bus	Ohjelmien väliseen kommunikointiin käytettävä rajapinta.
HTML	HyperText Markup Language. Avoimesti standardoitu kuvauskieli, jolla kirjoitetaan esimerkiksi nettisivuja.
LGPL-lisenssi	Lesser General Public License. Muutokset alkuperäiseen lähdekoodiin jaettava, mutta ei koske muuta tehtyä tuotetta.
SQLite	Relaatiotietokantajärjestelmä.
Tablet	Kämmentietokone
TCP/IP	Transmission Control Protocol / Internet Protocol. Useaan Internet-liikennöinnissä käytettävän tietoverkkoprotokollan yhdistelmä.
XML	Extensible Markup Language. Merkintäkieli datan säilytykseen ja siirtämiseen.

1 JOHDANTO

Tämä työ sai alkunsa Kuru Digital Creations Oy:n tarpeesta selvittää, kuinka suuria toimenpiteitä yrityksen täytyisi suorittaa, jotta yrityksen kehittämiä sovelluksia voitaisiin käyttää myös muilla laitteilla ja alustoilla, kuin yhtiössä oli alunperin suunniteltu.

Kuru Digital Creations Oy on yhtiö, joka tuottaa ja kehittää omia sovelluksiaan. Yhtiön sovellukset liittyvät pääasiallisesti eri urheilulajien valmennukseen ja pelien tilastojen esilletuontiin katsojille. Yhtiön tämän hetkinen painopiste on ollut jääkiekossa, mutta kaikkia sovelluksia on kehitetty niin, että ne ovat helposti muunneltavissa myös muiden lajien käyttöön.

Tässä työssä keskityttiin Kurun kehitteillä olevaan Scouting-sovellukseen, jonka tarkoituksena on olla kykyjenetsijöille heidän työtään helpottava työkalu. Scouting on kirjoitettu kokonaan Qt C++- ja QML-ohjelmointikielillä. Työn tavoitteena oli pystyä ajamaan Qt:llä kirjoitettua Scouting-sovellusta Android-laitteessa. Tämän lisäksi täytyi pystyä ottamaan kantaa Qt-koodin toimivuuteen Android-laitteessa, jotta myöhemmin olisi selvää, onko Android-osaajien hakeminen yritykseen tarpeellista, mikäli sovelluksia halutaan tehdä jatkossa enemmän Android-laitteille.

Työn aikataulussa pysymistä valvottiin Scrum-projektinhallintamenetelmällä. Työryhmän koko oli kuitenkin sen verran pieni, että kaikkia Scrumin tarjoamia työkaluja ei käytetty tämän projektin aikana. Aiheeseen perehdyttiin kuitenkin laajassa mittakaavassa.

2 SCOUTING

Mistä kaikki parhaat pelaajat tulevat? Kuka heidät löytää ja tuo muiden tietoisuuteen lajissa kuin lajissa? Scouttaajat eli kykyjenetsijät tekevät valtavat määrät työtä valokeilojen ulkopuolella, jotta juuri heidän edustamansa joukkue saa lupaavimmat nuoret ja muut noususssa olevat pelaajat omaan rivistöönsä (Malloy 2011, 3–4).

2.1 Ammattina scouttaus

Scouttausta eli kykyjenetsintää harjoitetaan karkeasti kahdessa osassa: amatööri- ja ammattipelaajien etsinnässä. Amatööreihin keskittyvät kykyjenetsijät keskittyvät nuoriin ja muihin pienemmissä liigoissa pelaaviin pelaajiin, kun taas ammattilaisiin keskittyvät toimivat liigojen sisällä, jossa pelaajat jo pelaavat ammatikseen. Amatööreihin keskittyvillä kykyjenetsijöillä on monesti pääsy paikallisten junioritiimien harjoituksiin ja muihin tapahtumiin, kun taas ammattilaisia etsivät kulkevat liigapeleissä ja joukkueiden harjoituksissa tekemässä havaintoja. Myös havainnot poikkeavat tietyllä tavalla. Amatööripelaajista haetaan uusia lahjakkuuksia ja teknillisesti hyviä pelaajia, joissa nähdään suurta potentiaalia kehittyä tarvittaessa vielä lisää vaadittavilla osa-alueilla. Ammattipelaajista haetaan monesti kykyä sopeutua tiettyyn rooliin joukkueessa kuin muutenkin joukkueeseen. Tietenkin myös ammattipelaajien osaamista ja kykyjen nousua ja laskua pidetään silmällä. (Malloy 2011, 19–21.)

Kykyjenetsijöiden työ voi olla uuvuttavaa ja haastavaa. Amatööripuolella voi joutua matkustamaan muutaman tunnin harjoitusten tai pelin takia valtavan pitkiä matkoja tai katsomaan päivässä useita tunteja nauhoitteita jo menneistä peleistä. Tutkittavia pelaajia voi olla tuhansia, joista ei tarvita ehkä kuin yksi tai kaksi. Lisäksi ei esimerkiksi ole helppoa löytää kykyjä ammattilaissarjan joukkueen neljännen kentän pelaajista, joilla ei ole kertynyt paljoa peliaikaa. Tuolla lyhyellä peliajalla joku pelaaja on kuitenkin voinut tehdä juuri kuten on käsketty, lukenut peliä hyvin ja reagoinut tilanteisiin nopeasti. Kummankin puolen kykyjenetsijöille on kuitenkin yhteistä se, että seurattaviin peleihin yleensä valitaan

muutamia pelaajia, joita pidetään silmällä. Pelaajista tehdään muistiinpanoja kaikennäköisten kiinnostuksen kohteena olevien asioiden tiimoilta pelin aikana. Pelin jälkeen voidaan vielä hakea lisätietoa menemällä pukuhuoneeseen ja haastattelemalla valmentajia ja pelaajia. Näin pelaajista saadaan valtava määrä tietoa raportteja varten, jotka lopulta merkitään jonnekin yhteiseen tietokantaan myöhempää tarkastelua varten. (Malloy 2011, 4–7.)

2.2 Scouting-sovellus

Scouting on Kuru Digital Creations Oy:n kehitteillä oleva pelaajatarkkailuun keskittyvä sovellus. Ohjelmalla pyritään helpottamaan kykyjenetsijöiden arkea heidän työtehtävissään, ja kaiken lähtökohtana onkin mahdollisimman helppo käytettävyys. Ohjelmalla optimoidaan jo olemassa olevia työkaluja paremmin kykyjenetsijöiden avuksi. Tätä tarkoittaa esimerkiksi nettiselain, joka tukee paremmin kykyjenetsijän tehtäviä ja toimii yhteen muun ohjelman osien kanssa. Kalenteri näyttää automaattisesti harjoituksiin merkittyjä tapahtumia. Lisäksi ohjelmalla pystyy tietenkin tekemään kykyjenetsijöiden varsinaista työtä eli pelaaja-arviointeja.

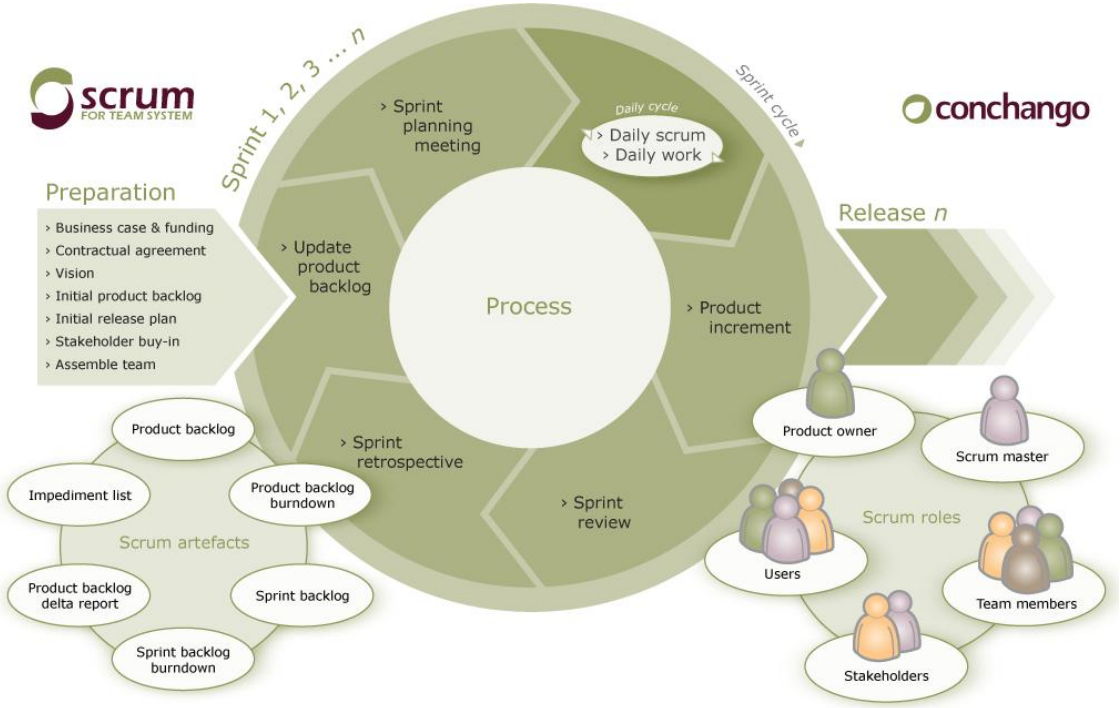
Kykyjenetsijä voi joko syöttää ohjelmaan pelaajan tiedot tai hakea saatavilla olevista järjestelmän tietokannoista pelaajia. Tämän jälkeen voidaan liittää pelaajia tiettyihin harjoituksiin ja pelaajiin arvioitavia kriteereitä. Arvioidessaan pelaajia kykyjenetsijä merkitsee tuloksia jo ennalta liitettyihin kriteereihin ja ohjelma tallentaa tulokset automaattisesti myöhempää tarkastelua varten. Kriteereitä voi toki lisätä kesken arvioinnin, mutta tarkoituksena on, että kun kriteerit ovat valmiina, itse arvioinnin aikana tulosten merkitsemiseen ei tarvitse käyttää kuin muutama painallus. Ohjelma tallentaa automaattisesti arvioijan tehtyyn arviointiin ja käyttäjän halutessa se voidaan tallentaa koko joukkueen yhteiseen tietokantaan, joka toimii hyvänä välineenä jakaa tietoa joukkueen kesken. Ohjelmasta löytyy myös budjetointi, jolla kykyjenetsijä voi arvioida pelaajiin tarvittavan rahan määrää ja annetun budjetin riittävyyttä.

Ohjelmaa on pyritty kehittämään niin, että se on helposti muokattavissa usealle eri lajille. Tämänhetkinen painoarvo on kuitenkin ollut jääkiekolla. Käyttöliittymä on suunniteltu kosketuskäyttöisille tablet-laitteille, mutta periaatteessa sen käyttö onnistuu millä tahansa PC:llä. Ohjelma on kuitenkin siis vasta kehitteillä, eikä se ole vielä jakelukelpoinen. Kuvassa 1 on Scouting-sovelluksen ensimmäinen kohdelaite WeTab.

KUVA 1. WeTab-tablet (WeTab. 2012)

3 SCRUM-PROJEKTINHALLINTAMENETELMÄ

Ketteriin projektinhallintamenetelmiin kuuluva Scrum mahdollistaa tuotekehityksen todellisen edistymisen helpon seurannan. Scrumin avulla voidaan viedä läpi useampia erilaisia kehitysprosesseja. Scrum soveltuu hyvin esimerkiksi jonkin kuluttajille tulevan ohjelmistopaketin vaiheistukseen tai vaikkapa yksittäiselle yritykselle tulevan web-sivuston toteutuksen läpivientiin. Scrum soveltuu monimutkaisiinkin kehitysprosesseihin, koska sen läpivienti perustuu jatkuvaan suunnitteluun eri vaiheiden edetessä eikä alussa tehtyjä ennustuksia lyödä lopullisesti lukkoon. (Lindström 2011.) Kuva 2 kuvaa kehitysprosessin eri vaiheita. Kuva selitetään tarkemmin tulevissa alaluvuissa.



KUVA 2. Scrum-prosessi (Bird 2006)

3.1 Yleistä

Scrum keskittyy tapaan, jolla projektia hallitaan ja viedään eteenpäin. Se ei ota kantaa matalan tason käytäntöihin. Sen sijaan se keskittyy projektin vaiheistamiseen ja jatkuvaan kontrollin pitämiseen projektin etenemisestä. Tästä syystä se toimiikin monesti hyvin johdantona asiakkaalle ketterien menetelmien maail-

maan. Kuten kuvassa 2 näkyy, Scrum-prosessikehitys rakentuu erimittaisten sprint-jaksojen ympärille. Sprint ja daily scrum ovat prosessin tärkeimmät jaksot. (Poimala – Tolvanen 2011.)

3.2 Valmistelut

Kuvassa 2 vasemmalla näkyvä laatikko "Preparations" eli valmistelut listaa kaikki kehitysprosessille vaadittavat lähtökohdat. Lista suomennettuna tarkoittaisi seuraavaa: "Liiketoiminta ja rahoitus, sopimuksenmukainen yhteisymmärrys, visio, alustava tuotteen kehitysjono, alustava julkaisusuunnitelma, sidosryhmän tuki ja koottu kehitystiimi".

3.3 Roolit

Kuvan 2 oikeassa alalaidassa on näkyvillä Scrum-projektin eri roolit. Projektin kannalta tärkeitä ovat tietenkin myös valmistuvan projektin käyttäjät ja sen takana oleva sidosryhmä, mutta varsinaiseen kehitysprosessiin ei lasketa mukaan kuin nämä kolme pääroolia: tuotteen omistaja, Scrum-mestari ja kehitystiimi.

3.3.1 Tuotteen omistaja

Tuotteen omistaja pitää hallussaan tuotteen kehitysjonoa. Hän priorisoi kehitysjonon kohtia ja valvoo kehitystiimin työn laatua. Tuotteen omistaja toimii myös välikätenä tiimin ulkopuolisille osapuolille, esimerkiksi tuotteen tilaajalle ja osakkeiden haltijoille. (Schaub 2010.)

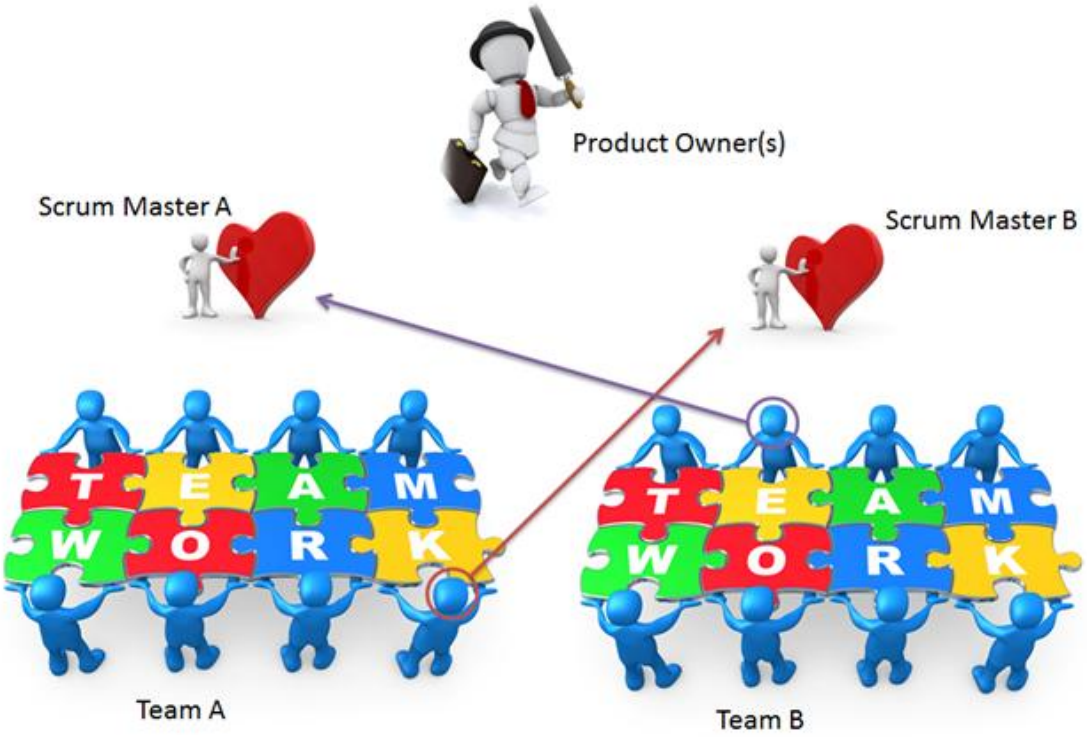
3.3.2 Scrum-mestari

Scrum-mestari toimii taas välikätenä tuotteen omistajan ja kehitystiimin välillä. Hän valvoo, että kehitystiimi toimii Scrumin ehtojen ja asetettujen sääntöjen mukaisesti. Hän myös auttaa kehitystiimiä mahdollisesti uhkaavien esteiden ratkomisessa, kuitenkaan puuttumatta sen tapaan toimia. Vaikka virallisesti Scrum-mestarin tehtäviin ei kuulu toimia kahvi- ja pitsalähettinä kehitystiimille, voivat nämä tehtävät edesauttaa ja nostaa moraalia kehitystiimissä esimerkiksi silloin, kun tehdään ylityötunteja. Scrum-mestarin ei tulisi olla tuotteen omistaja

eikä myöskään sen saman kehitystiimin jäsen, jolle hän toimii Scrum-mestarina, koska tästä voisi syntyä ristiriita esteisiin ja tehtäviin liittyvissä päätöksissä. (Schaub 2010.)

3.3.3 Kehitystiimi

Kehitystiimi on yleensä 4–9 henkilöä sisältävä ryhmä, joka toimii itsenäisesti projektin parissa. Kehitystiimi toimii kokonaisuutena, ja vaikka tehtäviä jaetaan henkilöiden erikoistumisen ja osaamisen mukaisesti, ovat kaikki tehtävät yhtä tärkeitä. Projektissa joko onnistutaan tai sitten epäonnistutaan yhdessä. (Schaub 2010.) Kuten kuva 3 esittää, Scrum-mestari voi toimia samaan aikaan kehitystiimin jäsenenä toisessa kehitystiimissä.



KUVA 3. Scrum-roolit (Schaub 2010)

3.4 Prosessit

Kuvan 2 Process-ympyrässä kuvataan yhden sprintin tapahtumia. Sprintit ovat Scrumin ydintapahtumia, jotka kestävät kuukauden tai sitä lyhyemmän ajan ja joiden pituus on aina sama koko kehityksen aikana. Sprintteihin kuuluu aina suunnittelupalaveri, päiväpalaveri, kehitystyö, sprinttikatselmointi, sprintin retrospektiivi ja kehitysjonon työstäminen. Sprintin jälkeen projektista tulisi olla tälle sprintille tuotettavaksi tarkoitetun tuoteversion julkaisukelpoinen versio valmiina. Seuraava sprintti käynnistetään välittömästä edellisen päätyttyä. (Schwaber – Sutherland 2011.)

Sprintti tulee viedä läpi ilman muutoksia sen lähtöasetteluihin. Sprintin aikana ei siis tehdä muutoksia sen tavoitteisiin, kehitystiimin koostumukseen tai sen laatutavoitteisiin. Sprintin sisällöstä voidaan kuitenkin käydä tarkentavaa keskustelua tuotteen omistajan ja kehitystiimin välillä, kun on opittu lisää ratkaistavista ongelmista. Sprintti myös keskeytetään hyvin harvoin ennen sen aikarajan päättymistä. Keskeytyksen voi tehdä vain tuotteen omistaja, mutta hänen näkemykseensä sen kannattavuudesta voivat vaikuttaa sidosryhmien, kehitystiimin ja Scrum-mestarin näkemykset. Keskeytys tapahtuu useimmiten vain silloin, kun sprintin tavoite muuttuu tarpeettomaksi. Tähän voivat vaikuttaa yrityksen suunnanmuutokset, markkinat tai teknologiset edellytykset. Keskeyttäminen on kannattavaa hyvin harvoin ja se tehdäänkin yleensä vain silloin, kun sprintin kannattavuus lakkaa. Myös keskeytetyn sprintin mahdolliset julkaisukelpoiset kehitysjonon kohdat katselmoidaan. Jos osa niistä on potentiaalisesti julkaisukelpoisia, kohdat yleensä arvioidaan uudelleen ja siirretään takaisin kehitysjonoon. (Schwaber – Sutherland 2011.)

3.4.1 Suunnittelupalaveri

Suunnittelupalaverissa koko Scrum-tiimi suunnittelee yhteistyössä sprintissä tehtävän työn. Suunnittelupalaveri saa kestää enintään kahdeksan tuntia kuukauden mittaisessa sprintissä ja sitä lyhyemmässä aikaa varataan suhteessa vähemmän. Suunnittelupalaveriin sisältyy kaksi eri osaa, jotka vastaavat kysy-

myksiin "Mitä tullaan toimittamaan alkavan sprintin tuoteversiossa?" ja "Miten tuoteversioon tarvittava työ voitaisiin toteuttaa?". (Schwaber – Sutherland 2011.)

Ensimmäisessä osassa tuotteen omistaja esittelee kehitystiimille kehitysjonon viimeisimmän version, jonka jälkeen kehitystiimi valitsee toiminnallisuudet tuotteen kehitysjonosta. Tämä valinta on täysin kehitystiimin päätettävissä, koska vain sen jäsenet tietävät aiemmasta kokemuksesta, mitä se pystyy toteuttamaan alkavassa sprintissä. Tämän jälkeen asetetaan sprintille tavoite, joka määräytyy toteutettavaksi valittujen kehitysjonon kohtien kautta. Tavoite antaa kehitystiimille selvän kuvan siitä, miksi tätä tuoteversiota lähdetään kehittämään. (Schwaber – Sutherland 2011.)

Toisessa osassa kehitystiimi suunnittelee, kuinka valmis tuoteversio toteutetaan. Kehitystiimi suunnittelee toiminnallisuuden sekä työt, jotka vaaditaan toimivan tuoteversion kehittämiseen. Työn määrä mukautuu kehitystiimin oman arvion mukaan, jonka se itse uskoo voivansa toteuttaa sprintin aikana. Valituista kehitysjonon kohdista ja suunnitelmasta, kuinka toteuttaa nämä kohdat, muodostuu tehtävälista, jonka mukaan kehitystiimi itse ohjautuu toteuttaakseen sprintin aikana vaadittavan työn. Tuotteen omistaja on usein tarkentamassa suunnittelun toisessa osassa kehitysjonon sisältöä sekä auttamassa mahdollisten kompromissien löytämistä. Mikäli kehitystiimi huomaa vielä tässä vaiheessa mahdollisia ongelmia, kuten työn liiallinen tai liian vähäinen työmäärä, se voi neuvotella tuotteen omistajan kanssa näiden ongelmien poissulkemiseksi. Kehitystiimi voi halutessaan kerätä neuvoja ulkopuolisiltakin henkilöiltä saadakseen esimerkiksi teknisiä tai liiketoiminnallisia neuvoja. Sprintin tavoite antaa hieman liikkumavaraa kehitystiimille siinä, kuinka tuoteversio toteutetaan. Suunnittelupalaverin jälkeen tulisi pystyä esittämään tuotteen omistajalle ja Scrummestarille se, kuinka tiimi aikoo työskennellä saavuttaakseen odotetun tuoteversion. (Schwaber – Sutherland 2011.)

3.4.2 Päiväpalaveri

Päiväpalaveri on kehitystiimin päivittäin pitämä kokoontuminen, jossa tarkastellaan tehtyä työtä edellisen päiväpalaverin jälkeen. Palaverin tarkoitus on tahdittaa keskeisimmät projektin vaiheet, tarkastella, onko projekti pysynyt aikataulussa ja ennustaa mitä keritään ja voidaan toteuttaa seuraavan 24 tunnin aikana ennen seuraavaa päiväpalaveria. Myös projektin kokonaistilanteen ja aikataulun tarkastelun kannalta päiväpalaverit ovat tärkeitä kehitystiimille. Päiväpalaverin paikka ja ajankohta sovitaan etukäteen samaksi jokaiselle palaverille, jotta sekaannuksia ei pääse tapahtumaan. (Schwaber – Sutherland 2011.)

Päiväpalaverissa jokainen kertoo vuorollaan, mitä on saanut aikaan edellisestä tapaamisesta, mitä aikoo tehdä ennen seuraavaa ja onko mahdollisia esteitä tiedossa. Scrum-mestarin tehtävänä on pitää huoli, että kehitystiimi pitää jokaisena päivänä päiväpalaverin ja että palaverit pysyvät lyhyinä, enintään 15 minuutin mittaisina. Scrum-mestari myös valvoo, ettei palavereihin tule kehitystiimin ulkopuolisia jäseniä. Päiväpalavereiden tarkoituksena ei ole raportoida, mutta se auttaa kehitystiimiä pysymään tilanteen tasalla niin, että se kykenee tarvittaessa selvittämään tilanteen päivittäin tuotteen omistajalle ja Scrum-mestarille. (Schwaber – Sutherland 2011.)

Päiväpalaverin avulla saadaan tunnistettua ja jopa poistettua kehityksen mahdollisia esteitä. Päiväpalaveri parantaa kommunikointia ja asiantuntemusta kehitystiimissä, mikä taas vähentää tarvetta ylimääräisille palavereille. Kehitystiimi voi kuitenkin pitää heti päiväpalaverin jälkeen tapaamisen, jossa suunnitellaan tarkemmin projektissa jäljellä olevaa työtä. (Schwaber – Sutherland 2011.)

3.4.3 Kehitystyö

Sprintin aikana valtaosa ajasta menee projektin työstämiseen. Jokaiselle kehitystiimin jäsenelle on annettu yleensä omaan erikoisosaamiseensa liittyvä työ, esimerkiksi ohjelmistoprojektissa ohjelmointi. Sprinttiin on voitu valita tehtäväksi esimerkiksi käyttöliittymän toteutus, jolloin yksi tai useampi tähän tehtävään valittu alkaa työstää käyttöliittymää. Suuremmissa projekteissa on monesti meneil-

lään monia eri tehtäviä, joita toteuttavat niihin suunnatut kehitystiimin jäsenet. (Poimala – Tolvanen 2011.)

3.4.4 Sprinttikatselmointi

Sprinttikatselmointi pidetään välittömästi sprintin loputtua. Katselmoinnissa tarkastellaan, mitä sprintin aikana on saatu aikaan, ja tarvittaessa sopeutetaan tuotteen kehitysjonoa. Katselmointi tapahtuu Scrum-tiimin ja sidosryhmien yhteistyössä. Katselmointi on kuukauden mittaisen sprintin jälkeen enintään neljän tunnin mittainen kokous ja kuukautta lyhemmillä sprinteillä katselmointiin käytetään suhteessa vähemmän aikaa. (Schwaber – Sutherland 2011.)

Sprinttikatselmuksessa kehitystiimi esittelee sprintin aikana aikaansaadun toteutuksen ja vastaa sitä koskeviin mahdollisiin kysymyksiin. Näin tuotteen omistaja saa käsityksen siitä, mikä projektissa on valmista ja mitä tulee vielä tehdä. Tuotteen omistaja myös kertoo kehitysjonon tilanteen ja arvioi, milloin tuotteen lopullinen valmistumisajankohta voisi olla. Kehitystiimi kokoaa ja arvioi sprintin aikana koetut onnistumiset ja ongelmat. Näiden tietojen perusteella saadaan hyvä kokonaiskuva projektin tämänhetkisestä tilanteesta. Koko tiimi pohtii yhdessä, mitä seuraavaksi voidaan ja kannattaa tehdä. Nämä tiedot antavat hyvän pohjan seuraavan sprintin suunnittelupalaverille. (Schwaber – Sutherland 2011.)

3.4.5 Sprintin retrospektiivi

Sprintin retrospektiivi pidetään sprinttikatselmoinnin ja seuraavan sprintin suunnittelupalaverin välissä, ja se antaa Scrum-tiimille mahdollisuuden tarkastella omaa työskentelyään edellisessä sprintissä. Tarkastelun tuotoksena voidaan saada aikaan kehitysprosessin parannuksia seuraavalle sprintille. Tarkastelussa otetaan huomioon seuraavat asiat: sprintin prosessin sujuvuus, ihmisten välisen yhteistyön onnistuminen, työkalujen toimiminen, prosessin eri osa-alueiden toimivuus ja mahdollisia parannuksia vaativat kohdat. Näiden tietojen perusteella luodaan suunnitelma parannuksille sen suhteen, kuinka toimia seuraavassa sprintissä tiiminä. Retrospektiivin tarkoituksena onkin hioa kehitystiimi

toimimaan mahdollisimman tuottavasti ja mielekkäästi Scrum-viitekehyksen sisällä. Retrospektiiviin käytetään enintään kolme tuntia kuukauden mittaisen sprintin aikana. (Schwaber – Sutherland 2011.)

3.4.6 Kehitysjonon työstäminen

Kehitysjonon työstäminen on toistuva, osa-aikainen prosessi, jossa tuotteen omistaja ja kehitystiimi lisäävät yhteistyössä yksityiskohtia tuotteen kehitysjonoon. Yksityiskohdat ovat esimerkiksi kehitysjonon kohdat ja niiden keskinäinen järjestys sekä työmääräarvioinnit. Vaikka työstäminen tapahtuu yleensä tuotteen omistajan ja kehitystiimin kesken, on kehitystiimillä yleensä riittävästi liiketoimintaosaamista, jotta se voi tarvittaessa työstää tuotteen kehitysjonoa myös itsenäisesti. Kaikki työmäärään liittyvät arvioinnit kehitystiimi tekee viime kädessä itse, koska vain työn tekevät ihmiset voivat antaa lopullisen työmääräarvion. Tuotteen omistaja voi kuitenkin auttaa kehitystiimiä ymmärtämään vaatimuksia ja tekemään niissä kompromisseja. Kehitysjonon työstämiseen käytetään yleensä enintään 10 % kehitystiimin kapasiteetista. (Schwaber – Sutherland 2011.)

3.5 Dokumentit

Kuvan 2 vasen alalaita kuvaa projektin mahdollisia dokumentteja. Scrum-projektin hallintaan voidaankin käyttää useita erilaisia dokumentteja, mutta yksinkertaisimmillaankin siihen kuuluu yleensä ainakin nämä kaksi dokumenttia: tuotteen kehitysjono ja sprintin tehtävälista. Tuotteen kehitysjonon on laatinut tuotteen omistaja, ja siitä löytyy tuotteen ominaisuuksien vaatimuslista tärkeysjärjestyksineen. Tästä listasta kehitystiimi valitsee sprintin suunnittelupalaverissa seuraavassa sprintissä toteutettavat kohdat ja suunnittelee niissä tarvittavat tehtävät. Näin muodostuu sprintin tehtävälista. (ScrumAlliance.)

Tuotteen kehitysjonon ja sprintin tehtävälistan lisäksi projekteissa on usein käytössä erilaiset edistymiskäyrät. Tuotteen edistymiskäyrä viittaa siihen, kuinka nopeasti kehitystiimi etenee kehitysjonon kohtien kanssa. Tämän avulla voidaan ennustaa julkaisun ajankohtaa tai päättää jonkin kehitysjonon kohdan poissul-

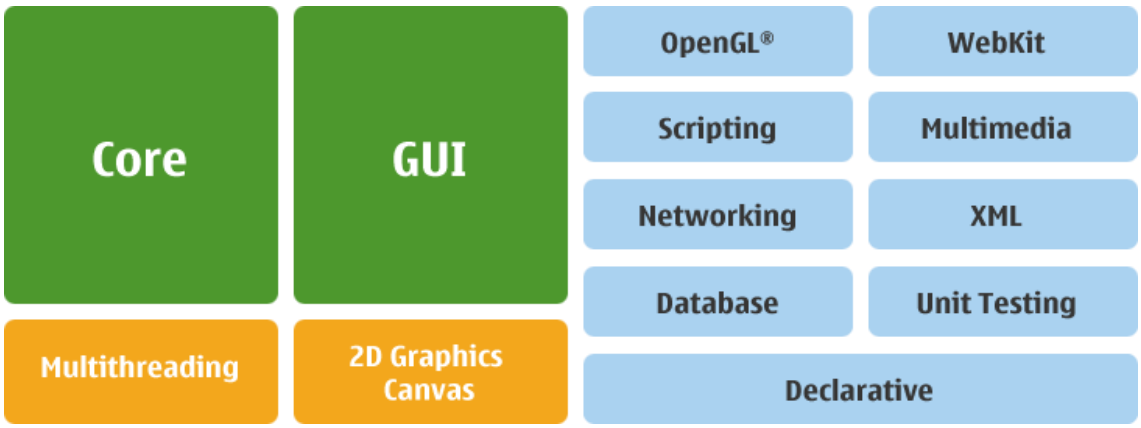
kemisesta, jos kehitystiimi ei etene toivotulla vauhdilla. Sprintin edistymiskäyrä kuvaa samalla tapaa sprintin tehtävien etenemistä. Sprintin edistymiskäyrä antaa monesti kehitystiimille viitteitä sprintin edistymisestä jo aivan sprintin alussa. Sen avulla voidaan tehdä ajoissa tarvittavia päätöksiä ongelmien korjaamiseksi, jos huomataan, että jokin tehtävä tarvitseekin oletettua enemmän työtä. (Scrum For Team System. 2010.)

Kuvan 2 tyyppisessä projektinhallinnassa on yllämainittujen dokumenttien lisäksi otettu mukaan myös tuotteen kehitysjonon tilat ja ongelmalista. Tuotteen kehitysjonon tilat näyttävässä listassa kuvataan kaikkien kehitysjonon kohtien nykyiset ja alkuperäiset tilat ennen tämänhetkisen sprintin alkua. Listassa näkyy myös kohtien kokonaistyömäärä sekä se, kuinka paljon tuosta määrästä on vielä jäljellä. Ongelmalista kuvaa, kuten nimikin sanoo, kaikki ennalta tiedettävät ongelmat, jotka voivat hidastaa tai pahimmillaan estää tehtävälistan kohdan toteutuksen. (Scrum For Team System. 2010.)

4 KÄYTETTÄVÄT TEKNOLOGIAT

4.1 Qt C++ ja Qt Quick

Qt on alustariippumaton ohjelmistokehitysympäristö, joka noudattaa periaatetta "kirjoita kerran, käännä missä tahansa". Qt-koodia voidaan kirjoittaa muun muassa kaikille näillä tunnetuille käyttöjärjestelmille: Windows, Linux ja Mac OS X. Lisäksi se tukee sulautettuja Linux-, Symbian- ja Windows CE -laitealustoja. (Qt Whitepaper 2012.) Erilaisista moduuleista koostuva Qt sisältää laajan valikoiman valmiita kirjastoja. Kirjastojen avulla edistyksellisten ja laitteistoriippumattomien sovelluksien kirjoittaminen on nopeaa. (Qt Modular Class Library 2012.) Kuva 4 kuvaa Qt-kirjastojen rakennetta.



KUVA 4. Qt-kirjastojen rakenne (Qt Modular Class Library. 2012)

Qt:n ohjelmointi tapahtuu hyvin pitkälle standardoidulla C++-ohjelmointikielellä. Se kuitenkin tuo hyvin laajoja lisäominaisuuksia, jotka nopeuttavat ja suoraviivaistavat ohjelmien kirjoittamista. Yksi Qt:n poikkeavimmista ominaisuuksista on ohjelman sisäisten tapahtumien käsittely, joka hoidetaan yhdistämällä ajossa syntyvät signaalit (signals) niitä käsitteleviin tapahtumankäsittelijöihin (slots). Tämä mekanismi mahdollistaa sen, ettei luokien tarvitse olla tietoisia toisistaan, mikä taas mahdollistaa hyvin useasti uudelleen käytettävien luokkien kirjoittamisen. Qt sisältää laajan valikoiman widgettejä eli erinäköisiä painikkeita ja muita

käyttöliittymäkomponentteja, joilla muodostetaan perustoiminnallisuus Qt:n graafisiin käyttöliittymiin. (Qt Whitepaper. 2012.)

Qt:lle on oma kehitysympäristö Qt Creator. Qt Creatorissa on kaikki tarvittavat ominaisuudet, joita kehitysympäristöltä voidaan odottaa. Siitä löytyy myös graafisten käyttöliittymien suunnitteluun tarkoitettu Qt Designer, jonka avulla graafisten käyttöliittymien tuottaminen on helppoa ja vaivatonta. Designerin työkalulaatikosta saa helposti vetämällä erilaisia widgettejä suoraan kehitteillä olevalle käyttöliittymäpohjalle. Desingerilla on helppo muuttaa muun muassa widgettien kokoa, paikkaa tai väriä ja se tallentaa kaikki tuotokset .ui-tiedostopäätteisiin tiedostoihin, joita voidaan muuttaa jäljestä niin Designerilla kuin manuaalisesti suoraan millä tahansa tekstieditorilla. Designer pystyy myös Qt 4.7:ssä esitellyn Qt Quickin graafisen käyttöliittymän luontiin ja muokkaamiseen. (Qt Whitepaper. 2012.)

Qt Quick on Qt:n avuksi suunniteltu sovelluskehitysmalli. Sen pääasiallinen käyttötarkoitus on helpottaa sulavien ja modernien käyttöliittymien tekoa kosketusnäytöllisiin mobiililaitteisiin. Ohjelmointi tapahtuu kuvailevalla (declarative) ohjelmointikielellä. QML on hyvin pitkälle JavaScriptin, HTML:n ja CSS:n yhdistelmä. Kuten edellä mainittiin, Qt Creatorin Designer mahdollistaa käyttöliittymien suunnittelun ja rakentamisen vetämällä suunnittelupohjalle työkalulaatikon valmiita QML-komponentteja tai itse tehtyjä komponentteja. Suunnittelija voi tuoda myös suoraan Photoshopilla tai GIMPillä suunniteltuja luonnoksia suoraan QML-tiedostoon. (Qt Quick Tooling Whitepaper. 2011.)

Nokia Oyj osti Qt:n alkuperäisen kehittäjän Trolltechin vuonna 2008 ja on jatkanut sen kehitystä siitä asti. (Reaktor – Qt-sovelluskehys 2011.) Qt:n lisenssit kuuluvat kuitenkin tänä päivänä kahdelle eri yritykselle. Nokia vastaa LGPL-lisensseistä, kun taas Digia Oyj hallitsee nykyisin kaupallista lisenssiä. (Koskenalho 2011.)

4.2 Android

Googlen omistama Android on noussut hyvin lyhyessä ajassa yhdeksi käytetyimmäksi käyttöjärjestelmäksi älypuhelimissa ja tablet-laitteissa. Google osti Androidia alun perin kehittäneen yhtiön Android Inc. vasta vuonna 2005 ja vuonna 2007 Androidista julkaistiin sen ensimmäinen versio. Googlen omien lausuntojen mukaan helmikuussa 2012 Android-laitteita aktivoitiin päivittäin jo 850 000 kappaletta ja Googlen ylläpitämässä Google Play -nettikaupassa Android-sovelluksia oli saatavilla 450 000 kappaletta. (Android (operating system) 2012.)

Androidin ydin perustuu Linux-ytimeen, jonka arkkitehtuuriin Google on tehnyt omia muutoksia. Androidissa ei myöskään ole Linuxista tuttua X-ikkunointijärjestelmää eikä se tue kaikkia GNU-kirjastoja, joista johtuen sille on hyvin vaikea tuoda jo olemassa olevia Linux-ohjelmia tai -kirjastoja. Android on kuitenkin lisensoitu avoimen lähdekoodin lisenssillä ja Google tarjoaa ilmaisen SDK:n Android-sovellusten kehittämiseen. Sovellukset kirjoitetaan pääasiallisesti Googlen räätälöimällä Javalla. (Android (operating system) 2012.)

4.3 Necessitas

Necessitas on koodinimi projektille, jonka tarkoituksena on tehdä Qt-porttaus Android-laitteisiin. Necessitaksen tarkoituksena on siis mahdollistaa Qt-koodin ajaminen Android-laitteessa. Necessitas pohjautuu Nokian Lighthouse-projektiin, joka jo liitettiin Qt 4.8:n jakeluversioon. Lighthouse mahdollistaa Qt:n porttauksen helpommin uusille sulautetuille laitteille. Necessitas-projektia vie eteenpäin KDE-yhteisö, ja sen tämän hetken viimeisin versio on Necessitas Alpha 3 update 4. Se ei siis vielä ole missään nimessä valmis, eikä sen toimivuus vielä ole taattu sataprosenttisesti. (Necessitas – Qt for Android 2012.)

Necessitas SDK:n mukana tulee kaikki, mitä tarvitaan Qt-koodin kirjoittamisen aloittamiseen Android-laitteelle. SDK:n matkassa tulee Qt SDK:sta tuttu Qt Creator sekä tarvittavat Java OpenSDK, Android NDK ja Apache Ant. Necessitas SDK on saatavilla niin Windowsille, Linuxille kuin myös Mac OS X:lle. Tä-

män lisäksi Android-laitteeseen on hyvä asentaa Ministro-sovellus, joka lataa automaattisesti Qt-sovelluksen vaatimat LGPL Qt:n jaetut kirjastot. (Sourceforge – Necessitas 2012.)

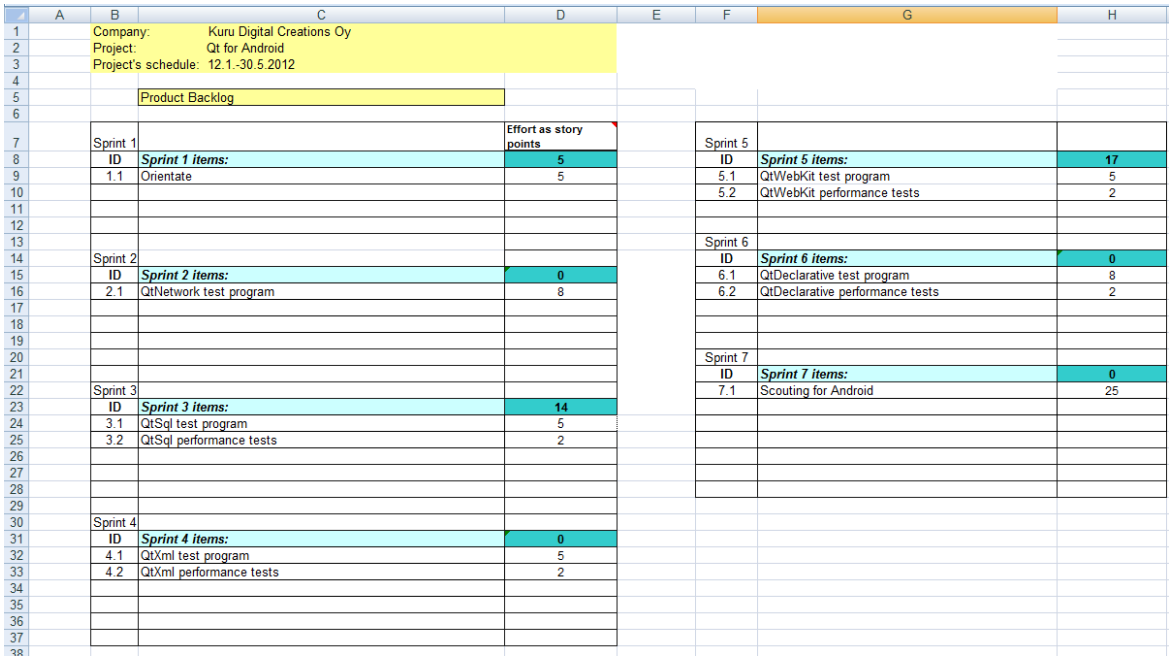
5 TOTEUTUS

Opinnäytetyön päätavoitteena oli siis saada ajettua Qt:lla kirjoitettua Scouting-sovellusta Android-laitteessa. Työtä päätettiin lähteä viemään eteenpäin jakamalla se erillisiin pienempiin projekteihin. Jokaisella pikku projektilla oli tarkoitus selvittää yksittäisen Qt-moduulin toimivuus Android-laitteessa. Tarkasteltavat moduulit rajattiin tärkeimpiin Scouting-sovelluksen kannalta. Myös moduulien sisältämien luokkien testaaminen rajattiin Scoutingin kannalta oleellisimpiin.

Työn aikana tehtävät pienemmät projektit kirjoitettiin alun perin täysin tavallisessa Qt-kehitysympäristössä, jonka jälkeen ne käännettiin Necessitas SDK:n avulla ja siirrettiin Android-laitteeseen ajoa varten. Tämän prosessin tarkoituksena oli selvittää, tarvitseeko erilaisiin Androidilla ajettaviin valmiisiin Qt-sovelluksiin tehdä muutoksia ja jos tarvitsee, niin minkälaisia, jotta sovellus toimii Androidilla. Käännöksen sulavuuden lisäksi tarkasteltiin suorituskykyä saman sovelluksen ajosta täysin Qt:tä tukevan ja Android-laitteen välillä.

5.1 Scrum-projektinhallintamenetelmän käyttö projektissa

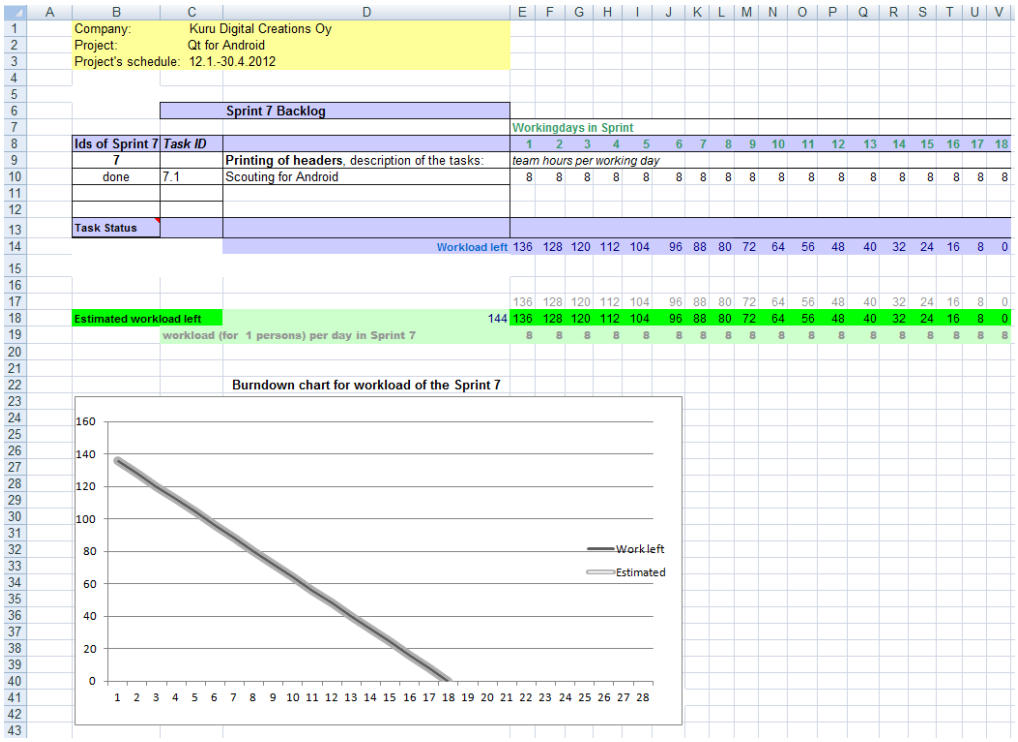
Työssä käytettiin Scrum-projektinhallintamenetelmää antamaan kehys tehtävälle työlle ja tarkastelemaan työn etenemistä aikataulussa. Aluksi luotiin tuotteen tehtävälista, joka hajotettiin seitsemään sprinttiin. Ohjeista poiketen sprinttien tehtävälistat suunniteltiin kaikille sprinteille valmiiksi heti aluksi. Kuvassa 5 on kuvankaappaus tuotteen kehitysjonosta.



Company:	Kuru Digital Creations Oy		
Project:	Qt for Android		
Project's schedule:	12.1.-30.5.2012		
	Product Backlog		
Sprint 1		Effort as story points	
ID	***Sprint 1 items:***	**5**	
1.1	Orientate	5	
Sprint 2			
ID	***Sprint 2 items:***	**0**	
2.1	QtNetwork test program	8	
Sprint 3			
ID	***Sprint 3 items:***	**14**	
3.1	QtSql test program	5	
3.2	QtSql performance tests	2	
Sprint 4			
ID	***Sprint 4 items:***	**0**	
4.1	QtXml test program	5	
4.2	QtXml performance tests	2	
Sprint 5			
ID	***Sprint 5 items:***	**17**	
5.1	QtWebKit test program	5	
5.2	QtWebKit performance tests	2	
Sprint 6			
ID	***Sprint 6 items:***	**0**	
6.1	QtDeclarative test program	8	
6.2	QtDeclarative performance tests	2	
Sprint 7			
ID	***Sprint 7 items:***	**0**	
7.1	Scouting for Android	25	

KUVA 5. Tuotteen kehitysjono

Työ vietiin läpi poikkeuksellisesti yhden hengen projektiryhmällä, ja näin ollen ei myöskään projektipalavereita projektin aikana pidetty. Projektin ja sprinttien etenemistä valvottiin sprinttien tehtävälistoilla, joihin piirtyivät kuvan 6 tapaa edistymiskäyrät automaattisesti työtuntien lisääntyessä.



KUVA 6. Sprint 7 Backlog

Kuvan 6 alalaidassa näkyy sprintin edistymiskäyrä. Vaaleampi harmaa käyrä kertoo sprintin alussa arvioidun edistymiskäyrän ja tummempi harmaa todellisen edistyksen. Tässä sprintissä nämä käyrät menivät päällekkäin, joten sprintti on onnistuttu viemään läpi arvioidussa ajassa.

Vaikka projekti oli tehtävälistaltaan suhteellisen yksinkertainen eikä eri sprinteissä ollut monia eri tehtäviä, oli Scrumista hyötyä työtuntien tarkkailussa. Se myös antoi alusta asti selvän suunnan projektille, jotta sitä voitiin viedä koko ajan eteenpäin alkuperäisten suunnitelmien mukaisesti.

5.2 Necessitasin toimivuus ja suorituskyky

Necessitas SDK:n toimivuutta lähdettiin tutkimaan Scouting-sovellukselle tärkeiden osa-alueiden kannalta, jotka jaettiin omiin pieniin projekteihin. Scouting käyttää järjestelmän oman datan lähettämiseen ja vastaanottamiseen TCP/IP-protokollaa, joten yhdeksi kohdaksi valittiin Qt:n QtNetwork-moduuli. Erinäköisen tiedon tallentamiseen käytetään SQLite-tietokantoja ja XML-dokumentteja,

joiden toiminnallisuus toteutetaan Qt:ssa QtSql- ja QtXml-moduuleilla. Scoutingin käyttöliittymä on toteutettu kokonaan QML:llä, joten QtGui-moduulin sijaan tarkempaan tarkasteluun otettiin QtDeclarative-moduuli. QtGuita käytettiin kuitenkin muiden projektien käyttöliittymien toteutukseen, koska se on hyvin tehokas yksinkertaisia käyttöliittymiä tehdessä. Näiden lisäksi Scoutingista löytyy QtWebKitillä toteutettu selain, joten myös tälle moduulille tehtiin yksinkertainen testaus perustoiminnoista.

Projektissa oli käytössä suorituskyvyltään hyvin eritasoiset laitteistot Qt:ta natiivisti tukevan ja Android-laitteen välillä, mistä johtuen suorituskyvyn mittaukset jäivät lähinnä silmämääräisiksi. Qt:ta natiivisti tukevan ja Scoutingin alkuperäisenä kohdelaitteena toimi WeTab-tablet. Android-laiteena toimi ASUS Eee Pad Transformer TF101. Tarkemmat laitteistotiedot ovat liitteessä 1.

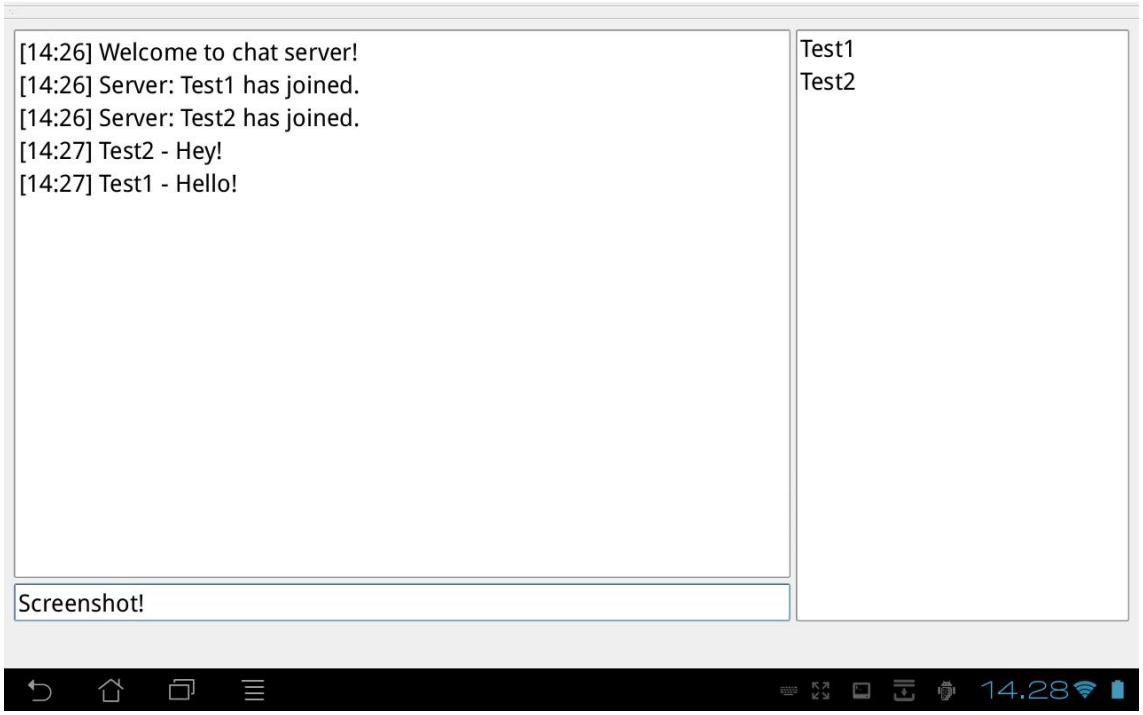
Projekteilla haluttiin nimenomaan selvittää tarvittavat muutokset jo valmiiden Qt-sovellusten kanssa, joten Necessitasta käytettiin vasta, kun haluttiin ajaa sovellusta Android-laitteessa. Kun Necessitas SDK:n matkassa tulleella Qt Creatorilla aukaistiin haluttu Qt-projekti, havaittiin sen lisäävän automaattisesti projektin matkaan Android-laitetta varten tarvittavat tiedostot. Tämän jälkeen kohdelaitteeseen tarvittiin Ministro-sovellus, joka lataa sovelluksen vaatimat Qt-kirjastot automaattisesti laitteeseen. Ministron saa ladattua ja asennettua Google Play -sovelluskaupasta ilmaiseksi. Necessitas SDK:n matkassa tulee myös Android emulaattori, jota voidaan käyttää koodin testaamiseen ilman fyysistä laitetta. Mikäli fyysinen laite on kuitenkin kytkettynä USB:llä PC:hen, jossa kehitysympäristöä käytetään, ajaa Necessitas sovellusta automaattisesti laitteessa "Run"-komennon saatuaan.

5.2.1 QtNetwork-moduuli

Qt:n TCP/IP-ominaisuuksia testatakseen Android-laitteessa tehtiin pieni keskusteluohjelman hyödyntäen Qt:n QTcpSocket-luokkaa. Jotta saadaan erilaiset verkossa kommunikoinnin mahdollistamat luokat käyttöön, täytyy Qt-projekti linkittää QtNetwork-moduulia vasten. Tämä tapahtuu lisäämällä projektin .pro-

tiedostoon rivi "QT += network". Tätä kokeilua varten ei tarvittu moduulin muita luokkia Android-laitteeseen tulevaa asiakasohjelmaa varten, mutta asiakasohjelmaa varten tehtiin myös palvelinohjelma, joka hyödynsi QTcpServer-luokkaa. Tätä ohjelmaa ei kuitenkaan ajettu Android-laitteessa.

Kuvassa 7 näkyy kuvankaappaus aikaansaadusta keskusteluohjelmasta Android-laitteessa. Ohjelma ottaa yhteyden tehtyyn palvelimeen, jonka jälkeen palvelin avaa socketin asiakasohjelmaa varten. Palvelimelle voi liittyä rajaton määrä asiakasohjelmia, ja kaikki liittyneet käyttäjät näytetään asiakasohjelmassa oikealla näkyvässä listassa. Kun käyttäjä lähettää viestin alhaalla näkyvän tekstikentän avulla, se välitetään palvelimen kautta kaikille asiakasohjelmille viestinäkymään.



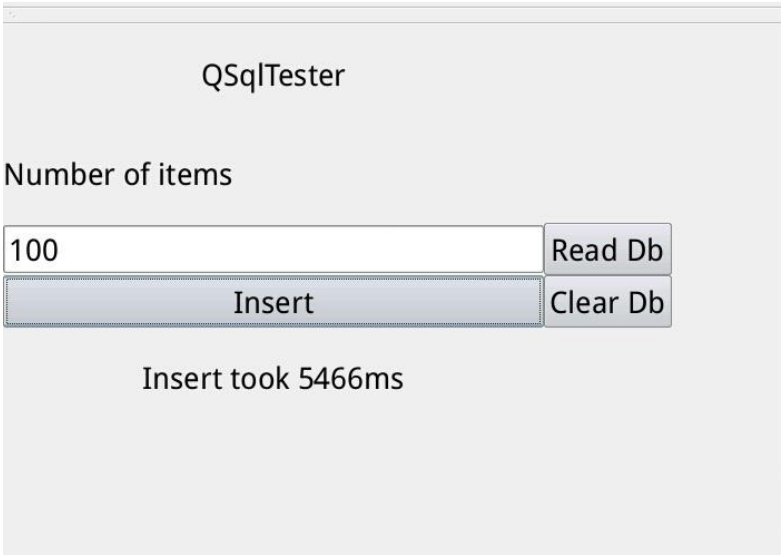
KUVA 7. Yksinkertainen keskusteluohjelma

Asiakasohjelman havaittiin toimivan suoraan Android-laitteessa. Android-käännöksen vuoksi ei tarvinnut tehdä muuta kuin aukaista projekti Necessitas SDK:n matkassa tulleella Qt Creatorilla ja painaa "Run"-painiketta.

5.2.2 QtSql-moduuli

Scouting-sovellus hyödyntää SQLite-tietokantoja tiedon tallennukseen, ja koska tärkeää dataa voi kertyä hyvinkin suuria määriä, ei sitä ole varaa kadottaa tai sen käsittelyssä ei saa kestää liian pitkiä aikoja. Qt:n QtSql-moduuli tuo tarvittavat luokat SQLite-tietokantakäsittelyjä varten ja projektin voi linkittää sitä vasten lisäämällä projektin .pro-tiedostoon rivin "QT += sql". Tällä kokeella haluttiin testata Qt-projektin yksinkertaisimpien SQLite-tietokantakomentojen toimivuus Android-laitteessa sekä nähdä, tapahtuuko tietokannan kirjoitus- tai lukunopeuksille suurempia hidasteluita.

Kuvassa 8 näkyy kuvankaappaus tehdystä ohjelmasta. Ohjelman tekstikenttään syötetään haluttu määrä kirjoitettavista riveistä. Ohjelman koodiin on ennalta määrätty string-tyyppinen muuttuja, joka kirjoitetaan kullekin riville. Insert-painiketta painettaessa ohjelma kirjoittaa stringin niin monesti kuin käyttäjä on halunnut ja ilmoittaa alla kirjoittamiseen kestäneen ajan. Read Db -painike lukee koko tietokannan ohjelman omaan listaan ja ilmoittaa siihen kuluneen ajan. Clear Db -painike tyhjentää tietokannan.



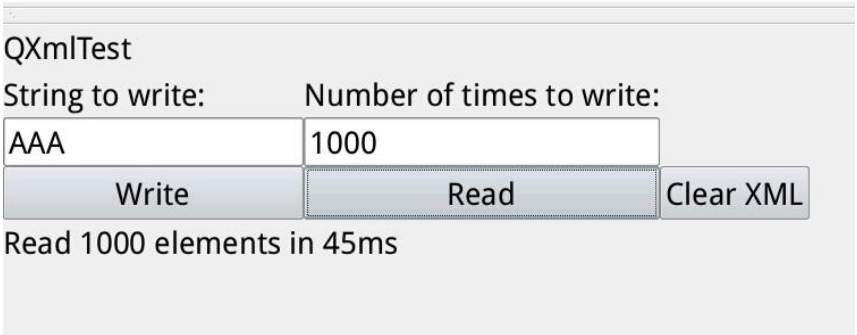
KUVA 8. QSqlTester

Ohjelma toimi Androidilla suoraan ilman mitään ongelmia. Kirjoituksiin kestäneiden aikojen vertailussa Android-laitteen ja muiden Qt:ta natiivisti tukevien laitteiden välillä voitiin todeta, että kirjoitusnopeudet olivat verrannollisia laitteiden massamuistin kirjoitus- ja lukunopeuksien kanssa, eikä Necessitas-käännöksestä siis näin ollen aiheutunut hidasteluita näissä toiminnoissa.

5.2.3 QtXml-moduuli

Scouting tallentaa dataa myös XML-dokumentteihin, ja myöskään näiden datojen tallennuksessa ei tule olla ongelmia. Qt tarjoaa XML-dokumenttien käsittelyyn moduulin QtXml. Moduuli otetaan käyttöön lisäämällä projektin .pro-tiedostoon "QT += xml". Myös tässä kokeilussa haluttiin varmistua tavallisten XML-komentojen toimivuudesta ja siitä, ettei kirjoitus- ja lukunopeuksissa ole suuria poikkeavuuksia.

Kuvassa 9 on kuvankaappaus tehdystä ohjelmasta. Ohjelmalle annetaan stringi, joka halutaan kirjoittaa, ja kirjoitettava stringien määrä. Write-painiketta painettaessa, ohjelma kirjoittaa käyttäjän määräämän stringin XML-tiedostoon niin monesti kuin käyttäjä on halunnut. Tämän jälkeen ohjelma ilmoittaa kirjoitukseen kestäneen ajan. Read-painike lukee XML-tiedoston elementit ohjelman lista-olioon ja ilmoittaa siihen kuluneen ajan. Clear XML -painike tyhjentää XML-dokumentin.



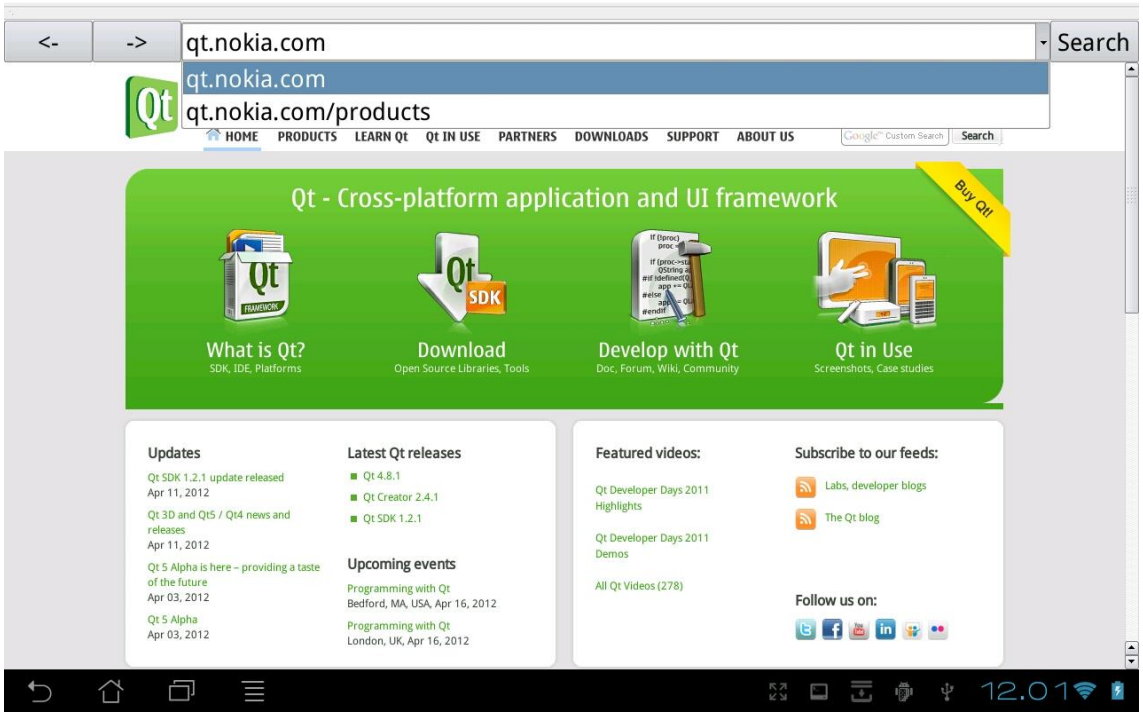
KUVA 9. QXmlTest

Myös tämä ohjelma toimi suoraan Android-laitteessa ja kirjoitus- ja lukunopeuksista voitiin todeta niiden jälleen olevan massamuistin nopeuksista riippuvaisia.

5.2.4 QtWebKit-moduuli

Qt tuo hyvin helppokäyttöisen QtWebKitin Qt-projekteille. WebKitin avulla saa minkä tahansa WWW-sivun näkyviin muutamalla koodirivillä. Moduuli otetaan käyttöön kuten muutkin moduulit lisäämällä .pro-tiedostoon "QT += webkit". QtWebKit toimii Scouting-sovelluksen selaimen pohjana, ja näin ollen myös sen toimivuus haluttiin varmistaa tällä yksinkertaisella projektilla.

Kuvassa 10 on Android-laitteessa toimiva QtWebKit-pohjainen selain. Selaimeen tehtiin vain perusominaisuudet: halutun WWW-sivun syöttö, edellisten syötettyjen sivujen listaaminen, edellinen sivu ja seuraava sivu. Selain toimi ilman ongelmia myös Android-laitteessa, eikä sen käytössä tai sivujen latauksessa havaittu mitään poikkeavia hidasteluita.

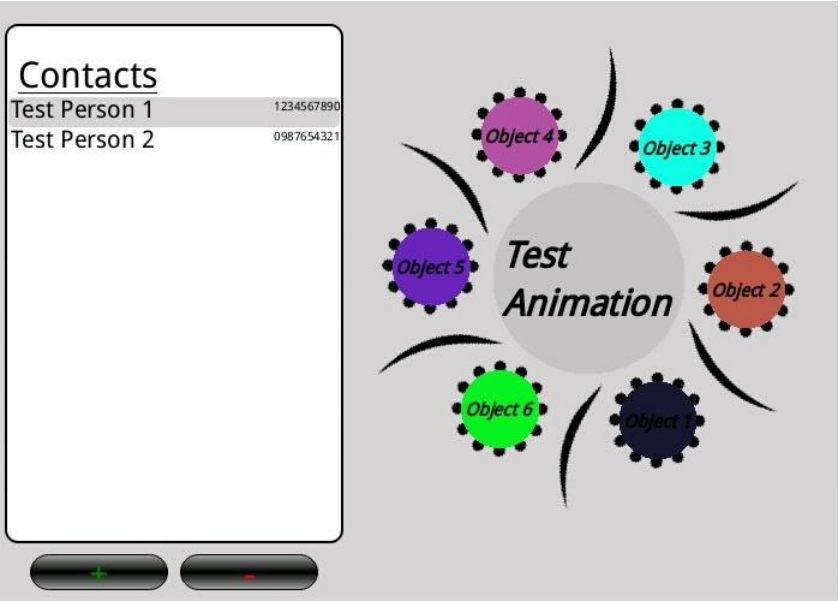


KUVA 10. QtWebKitillä toteutettu yksinkertainen selain

5.2.5 QtDeclarative-moduuli

QtDeclarative-moduuli mahdollistaa Qt Quick-sovelluskehityksen käyttämisen Qt-projekteissa. Scoutingin kaikki käyttöliittymät on toteutettu Qt Quickillä, joten on Android-käännöksen kannalta elinehto, että QtDeclarative toimii myös Androidilla. Qt Quickin on myös sanottu olevan laitteelta hyvin suorituskykyä vaativa, joten sen toimivuutta yritettiin myös pitää silmällä sen sulavuuden kannalta. QtDeclarative-moduuli otetaan käyttöön lisäämällä projektin .pro-tiedostoon "QT += declarative".

Kuvassa 11 on tässä kokeilussa aikaan saadun ohjelman päänäkymä. Ohjelmassa vasemmalla on näkyvillä QML:n ListView-elementti. Sitä esiintyy paljon Scouting-sovelluksesta, ja tästä syystä sitä haluttiin testata myös tässä ohjelmassa. Lisäksi kuvassa näkyy kaksi projektissa tehtyä elementtiä. Vasemmalla alhaalla näkyvät painikkeet ja oikealla näkyy animaatio, jonka liikkumista ei tietenkään havaita kuvasta. Listaan on mahdollista lisätä yhteystietoja painamalla "+"-painiketta. Kuvaan tulee ponnahdusikkuna, jossa on kentät nimelle ja numerolle. "-"-painike poistaa listasta valitun yhteystiedon. Animaatiossa kaikki laidalla olevat objektit kiertää vastapäivään keskimmäisen, isomman ympyrän ympärillä. Keskellä olevaa ympyrää painettaessa ohjelma arpoo pienemmille palloille uudet värit.



KUVA 11. QtDeclarativeTest

QML-tiedostoihin on mahdollista kirjoittaa JavaScriptillä myös paljon logiikkaa, mutta monimutkaisemmat Qt-sovellukset vaativat aina pohjalle C++-rakenteen. Vaikka tämän projektin toiminnot olisi voinut toteuttaa kokonaan JavaScriptillä QML-tiedostoihin, toteutettiin pienempien pallojen uusien värien arpominen C++-koodin puolella. Näin saatiin testattua, miten Qt:n signal/slot-sisäisten tapahtumien käsittely toimii C++- ja QML-koodin välillä myös näissä olosuhteissa.

Tämäkin testi osoitti, että Necessitas toimii kuten pitää. Ohjelman toiminnallisuudessa ei ollut minkäänlaisia ongelmia, eikä suorituskyvyssäkään havaittu moitteita. Alun perin raskaaksi epäilty animaatio pyöri näytöllä erittäin sulavasti ohjelman ajon aikana.

5.3 Scouting Android-laitteessa

Jo projektin alussa oli selvää, että Scoutingin kaikki ominaisuudet eivät tulisi toimimaan suoraan. Scoutingin kalenteri käyttää D-Busia muiden ohjelmien kanssa kommunikointiin, eikä Androidista löydy tukea tälle. Ongelma ei kuitenkaan ollut liian suuri, koska kalenteri voitaisiin myöhemmin toteuttaa ilman D-Busia. Tätä toteutusta ei kuitenkaan ollut tarkoitus tehdä tässä projektissa, joten

kalenteri otettiin pois käytöstä projektin ajaksi. Varsinaisena ratkaistavana ongelmana pidettiin alusta asti itse tehtyjen liitännäisten lataamista ajoon. Tästä kehittyikin yllättävänkin iso ongelma, jonka ratkaisuunkin meni yllättävän paljon aikaa. Onneksi aikaa varattiinkin jo alusta asti riittävästi, vaikka sitä aluksi epäiltiinkin olevan jopa liikaa. Ongelmat eivät kuitenkaan olleet loppujen lopuksi isoista seikoista kiinni.

Kun projektia ajetaan Necessitas SDK:n kautta, se luo projektista Android-laitetta varten paketin, joka kopioi kaikki kirjastot lib-hakemiston alle, joka sijaitsee ohjelman asennushakemiston juuressa. Tämän lisäksi kirjastoja ei tuotu ohjelman käyttöön automaattisesti, vaan liitteen 2 kaltaiset muutokset täytyi tehdä osaan Necessitaksen automaattisesti projektiin luomista tiedostoista. Ilmeisesti tämä oli tunnettu ongelma myös Necessitas-kehitystiimille, koska uusimman päivityksen olisi pitänyt poistaa tämä ongelma päivitystietojen mukaan. Tässä työssä ongelma esiintyi kuitenkin vielä myös päivityksen jälkeen.

Scouting muodostuu yhden QDeclarativeViewin ympärille tuoduista liitännäiskirjastoista, joka luodaan ohjelman main.cpp:ssä. Jotta näitä liitännäiskirjastoja voidaan tuoda ajoon, täytyy kutsua viewin engineä, jolle annetaan hakemistopolku, josta liitännäiskirjastoja löytyy. Kun jokaisen liitännäisen hakemistosta löytyy "qmldir"-niminen tiedosto, jossa ensimmäisellä rivillä lukee "plugin 'kirjaston_nimi'", lataa engine liitännäiskirjaston automaattisesti liitännäisen polun saatuaan. Koska Scouting ja sen kirjastot oli jo aiemmin rakennettu tätä tapaa tukevaksi ja koska Necessitas kopioi kaikki kirjastot saman hakemiston alle, työssä piti ainoastaan antaa QML enginelle liitännäisten poluksi tuo hakemisto seuraavaan tapaan: engine->addPluginPath(QDir::homePath()+"/../lib");. Tämän toimivuus edellytti myös kaikkien QML-tiedostojen ja kuvien lisäämistä Qt:n resource-tiedostoihin sekä ohjelmassa käytettävien polkujen muuntamista viittaamaan tiedostoihin resource-tiedostojen kautta. Jokaisella resource-tiedostolla tuli olla yksilölliset nimet. Näiden toimenpiteiden jälkeen Scouting käynnistyi ja toimi Android-laitteessa riittävällä tavalla tämän projektin kannalta.

Kuitenkin esimerkiksi palvelimeen liittyvät ohjelman osat jäivät testaamatta, koska sovellukselle ei ollut tämän projektin aikana saatavilla vielä palvelinta.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli selvittää Qt:n toimivuutta Androidilla. Työssä haettiin mahdollisia ongelmakohtia eri osa-alueilta pienempien työn aikana tehtyjen testiohjelmien avulla. Lopullisena haasteena oli saada toimimaan ohjelmakoodiltaan huomattavasti laajempi Scouting-sovellus Android-laitteessa ja saada aikaan johtopäätös, onko Qt-koodin toimivuus vielä riittävää Androidilla.

Alussa tehtyjen pienempien sovellusten toimivuus yllätti positiivisesti. Kaikki tuntui toimivan helposti ja hyvin Necessitas SDK:n avulla. Sovellukset toimivat Androidilla yhtä vaivattomasti kuin Linux-laitteilla ajettaessa. Scoutingin ongelmien ratkomisessa sen sijaan menikin hieman kauemmin. Ongelmia ei loppujen lopuksi ollut paljon ja ne eivät olleet edes suuria, mutta ne vaativat tietynlaista kokeilua ja näpertelyä. Ongelmia toikin paljon vielä hieman keskeneräinen Necessitas SDK, jota joutui muun muassa asentelemaan useampaan kertaan ja jopa hieman sen kehitysyhteisön ohjeista poiketen, jotta sen sai toimiaan.

Työn toteutuksessa Scrumin käyttö jäi ehkä hieman suppeaksi työryhmän koon takia, mutta työssä tehdyt tutkimukset Scrumista opettivat paljon lisää sen hyödyllisyydestä ja kätevyydestä suuremmissa projekteissa. Työn pienempiä sovelluksia tehtäessä oivalsi yhtä ja toista Qt:sta, miten jokin asia olikin helpompi toteuttaa eri tavalla kuin oli aikaisemmin tottunut tekemään. Myös Necessitas SDK:n käyttö tuli tutuksi, ja mikäli sama työ tulisi nyt tehdä uudelleen, tapahtuisivat asiat paljon sujuvammin ja nopeammin.

Työn aikana kävi selväksi, että Qt-koodia todellakin voi jo ajaa Androidilla, enkä itse näe ainakaan tämän projektin ja Scouting-sovelluksen kannalta mitään hyötyä kirjoittaa sovelluksia uudelleen Javalla Android-kirjastoja hyödyntäen. Vaikka Necessitas kärsii vielä pienistä ongelmista, se on jo käyttökelpoinen, eikä sen ensimmäinen toimivaksi todettu julkaistava versiokaan todennäköisesti enää kaukana ole.

LÄHTEET

Android (operating system). 2012. Saatavissa: http://en.wikipedia.org/wiki/Android_(operating_system). Hakupäivä 18.4.2012.

ASUS – Eee Pad Transformer TF101. 2012. Saatavissa: http://www.asus.fi/Eee/Eee_Pad/Eee_Pad_Transformer_TF101/. Hakupäivä 14.5.2012

Bird, C. 2006. Scrum for Team System - New Scrum Diagram. Saatavissa: http://consultingblogs.emc.com/colinbird/archive/2006/03/10/3058.aspx. Hakupäivä 12.4.2012.

Koskenalho, S. 2011. Digia ostaa Qt-ohjelmistojen kaupallisen lisensointi- ja palveluliiketoiminnan Nokialta. Saatavissa: http://www.digia.com/fi/Digia/Yritys/Lehdisto/2011/Digia-ostaa-Qt-ohjelmistojen-kaupallisen-lisensointi--ja-palveluliiketoiminnan-Nokialta/. Hakupäivä 19.4.2012.

Lindström, J. Scrum. 2011. Saatavissa: http://reaktor.fi/osaaminen/scrum/. Hakupäivä 16.4.2012.

Malloy, S. 2011. The art of Scouting. How the hockey experts really watch the game and decide who makes it. Wiley. Ontario Kanada.

Necessitas – Qt for Android. 2012. Saatavissa: http://qt-project.org/wiki/Necessitas. Hakupäivä 21.3.2012.

Poimala, S. – Tolvanen, P. 2011. Ketteryys haltuun: Scrum pähkinänkuoressa. Saatavissa: http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-scrum-pahkinankuoressa/. Hakupäivä 16.4.2012.

Qt Modular Class Library. 2012. Saatavissa: http://qt.nokia.com/products/library. Hakupäivä 14.5.2012.

Qt Quick Tooling Whitepaper. 2011. Saatavissa: http://qt-project.org/wiki/QtQuickToolingWhitepaper. Hakupäivä 20.3.2012.

Qt Whitepaper 2012. Saatavissa: http://qt-project.org/wiki/QtWhitepaper. Hakupäivä 20.3.2012.

Reaktor – Qt-sovelluskehys. 2011. Saatavissa: http://reaktor.fi/osaaminen/qt-sovelluskehys/. Hakupäivä Hakupäivä 19.4.2012.

Schaub W. 2010. Basics of scrum ... Part 1: What are the scrum roles and how do the fit in with the Rangers? Saatavissa: http://blogs.msdn.com/b/willy-peter_schaub/archive/2010/04/24/basics-of-scrum-part-1-what-are-the-scrum-roles-and-how-do-the-fit-in-with-the-rangers.aspx. Hakupäivä 12.4.2012.

Schwaber, K. – Sutherland, J. 2011. The Scrum Guide. Saatavissa: http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20FI.pdf. Hakupäivä 12.4.2012.

Scrum Alliance. Scrum: The Basics. Saatavissa: http://www.scrumalliance.org/pages/what_is_scrum/. Hakupäivä 18.4.2012.

Scrum for Team System. 2010. Guidance. Saatavissa: http://www.scrumforteamsystem.co.uk/ProcessGuidance/. Hakupäivä 12.4.2012.

Sourceforge – Necessitas. 2012. Saatavissa: http://sourceforge.net/p/necessitas/home/necessitas/. Hakupäivä 21.3.2012.

WeTab. 2012. Saatavissa: http://wetab.mobi/en/. Hakupäivä 14.5.2012

WeTab, tablet

Käyttöjärjestelmä:	WeTab OS, perustuu MeeGoon
Näyttö:	11,6" (29,5cm), Resoluutio: 1366 x 768, kosketusnäyttö
Prosessori:	1,66 GHz Intel® Atom™ N450 Pineview
Muisti:	1 GB
Massamuisti:	16 GB tai 32 GB riippuen mallista. Tukee SDHC-kortteja
Wi-Fi:	On
Bluetooth:	On
3G:	Riippuen mallista

(WeTab 2012.)

ASUS Eee Pad Transformer TF101, tablet + näppäimistötelakka

Käyttöjärjestelmä:	Android 4.0, Ice Cream Sandwich (päivitetty jäljestä)
Näyttö:	10,1". Resoluutio: 1280 x 800, kosketusnäyttö
Prosessori:	NVIDIA® Tegra™ 2 1,0 GHz dual-core
Muisti:	1 GB
Massamuisti:	16 GB tai 32 GB riippuen mallista. Tukee SD-kortteja
Wi-Fi:	On
Bluetooth:	On
3G:	Ei

(ASUS – Eee Pad Transformer TF101 2012.)

libs.xml alkuperäisenä muotona:

```
<?xml version='1.0' encoding='utf-8'?>
<resources>
   <array name="qt_libs">
      <item>QtCore</item>
      <item>QtGui</item>
      <item>QtNetwork</item>
      <item>QtScript</item>
      <item>QtSql</item>
      <item>QtSvg</item>
      <item>QtXmlPatterns</item>
      <item>QtDeclarative</item>
   </array>
   <array name="bundled_libs"/>
</resources>
```

Muutoksien jälkeen:

```
<?xml version='1.0' encoding='utf-8'?>
<resources>
   <array name="bundled_libs_fixed">
     <item>Item0</item>
     <item>Item1</item>
     <item>Item2</item>
   </array>
   <array name="qt_libs">
      <item>QtCore</item>
      <item>QtGui</item>
      <item>QtNetwork</item>
      <item>QtScript</item>
      <item>QtSql</item>
      <item>QtSvg</item>
      <item>QtXmlPatterns</item>
      <item>QtDeclarative</item>
   </array>
   <array name="bundled_libs"/>
</resources>
```

Item0, 1 ja 2 kuvaavat kirjastoja, joita halutaan tuoda mukaan. Esimerkiksi libEsimerkki.so laitettaisiin muotoon: <item>Esimerkki</item>.

AndroidManifest.xml:ssä muutosta vaativa kohta:

```
</intent-filter>
        <meta-data android:name="android.app.qt_libs_resource_id" an
            droid:resource="@array/qt_libs"/>
        <meta-data android:name="android.app.bundled_libs_resource_id" an
            droid:resource="@array/bundled_libs"/>
```

Muutettuna:

```
</intent-filter>
        <meta-data android:name="android.app.qt_libs_resource_id" and
            roid:resource="@array/qt_libs"/>
        <meta-data android:name="android.app.bundled_libs_resource_id" and
            roid:resource="@array/bundled_libs_fixed"/>
```